Challenges of API Documentation from a Provider Perspective and Best Practices for Examples in Public Web API Documentation

Gloria Bondel, Arif Cerit and Florian Matthes
Faculty of Informatics, Technical University of Munich, Boltzmannstrasse 3, Garching, Germany

Keywords: API Documentation, Web APIs, Challenges, Best Practices.

Abstract: Developers frequently have to learn new Web APIs provided by other teams or organizations. Documentation, especially code examples, supports learning and influences the consumers' perception of an API. Nevertheless, documentation repeatedly fails to address consumers' information needs. Therefore, we identify four major challenges of creating and maintaining public Web API documentation from a provider perspective which are unknown customer needs, the difficulty of balancing the coverage of varying information needs and keeping documentation concise, the high effort of creating and maintaining documentation, and missing internal guidance and governance for creating API documentation. In addition, we derive 46 best practices candidates for code examples as part of Web API documentation from literature and 13 expert interviews. Moreover, we evaluate a subset of eight of these candidates in the context of the Web API documentation for a public GraphQL API in a case study with 12 participants. As a result, we validate the analyzed eight best practices candidates to be best practices for public Web API documentation.

1 INTRODUCTION

Nowadays, learning new APIs provided by other teams or organizations is a common task for developers (Meng et al., 2018; Meng et al., 2019; Glassman et al., 2018). Hence, developers often face the challenge of evaluating the suitability of an API for solving a specific problem (Meng et al., 2018). Also, developers often have to figure out how to use an API efficiently to solve a specific problem (Meng et al., 2018). Information necessary to accomplish these tasks can comprise knowledge about how domain concepts map to API elements, what use cases the API supports, how different requests affect resource consumption, and how the API reports errors (Robillard and DeLine, 2011; Meng et al., 2018).

API providers use documentation to transfer this kind of information to developers in other teams or organizations. Therefore, documentation is a crucial learning resource for API consumers (Robillard, 2009; Lethbridge et al., 2003; McLellan et al., 1998). Moreover, API consumers perceive documentation-related issues, e.g., errors or missing information, as a significant impediment when learning APIs (Meng et al., 2018; Robillard and DeLine, 2011; Robillard, 2009). Thus, the success of a public API can depend on the documentation's ability to meet the consumers' information needs (Meng et al., 2018).

Previous research reveals the vital role of code examples in API documentation (Nykaza et al., 2002; Meng et al., 2018; Meng et al., 2019; Robillard, 2009; Ko et al., 2007; Nasehi and Maurer, 2010; McLellan et al., 1998; Meng et al., 2020; Jeong et al., 2009; McLellan et al., 1998). For instance, examples represent entry points to learning a new API (Meng et al., 2018; Meng et al., 2019) as well as to solve specific problems (Meng et al., 2019; Nykaza et al., 2002; Meng et al., 2018). Also, developers perceive examples as more informative and easier to understand compared to textual descriptions and they convey a feeling of how to best use an API (Meng et al., 2018).

Even though the importance of documentation and especially examples is well researched, consumers report that documentation only rarely meets their needs (Meng et al., 2018). Inspecting literature reveals that challenges related to API documentation are investigated only from a consumer point of view, e.g., by (Robillard and DeLine, 2011) and (Meng et al., 2018). In comparison, to the best of the authors' knowledge, there are no previous studies identifying challenges that API providers face when designing and maintaining Web API documentation for API consumers external to the API providers team or organization.

Secondly, if documentation entails examples,

Bondel, G., Cerit, A. and Matthes, F.
Challenges of API Documentation from a Provider Perspective and Best Practices for Examples in Public Web API Documentation.
DOI: 10.5220/0011089700003179
In *Proceedings of the 24th International Conference on Enterprise Information Systems (ICEIS 2022) - Volume 2*, pages 268-279
ISBN: 978-989-758-569-2; ISSN: 2184-4992

these examples have to meet specific quality criteria to unlock their potential and support the learnability of APIs (Meng et al., 2018; Robillard and DeLine, 2011; Robillard, 2009; Nykaza et al., 2002). Therefore we aim to answer the following two research questions:

RQ1: What challenges do API providers face when designing and maintaining public Web API documentation?

RQ2: What are best practices for creating code examples as part of public Web API documentation?

We conducted interviews with 13 experts to answer the first research question. As a result, we identify four significant challenges that API providers face when creating public Web API documentation. To answer the second research question, we first identify 46 best practices candidates for code examples in public Web API documentation from literature and the same interview series used to answer the first research question. Finally, we evaluate if a subset of the identified best practices candidates influences consumers' perception and productivity in a case study. As a basis for the case study, we create two versions of documentation for an existing public Web API that differ regarding the implementation of eight best practices candidates. Afterward, we observed 12 professional software developers while solving a set of defined tasks using one of the two documentation versions to evaluate the effect of the eight best practices candidates in the context of Web APIs. The results show that the implemented best practices candidates positively influence the consumers' perception of the API, and thus we validate the candidates to be best practices.

The contribution of this research paper is threefold. First, we present four central challenges of creating public Web API documentation from a provider perspective. Each of these challenges provides a starting point for future research. In addition, we present a list of best practices candidates for code examples in public Web API documentation and validate a subset of eight of these candidates to be best practices. Practitioners can use these best practices to guide the creation of API documentation. Also, these best practices can guide the design of tools that support or automate the generation of Web API documentation. In addition, future research can investigate the effect of other best practices candidates that we did not analyze as part of the presented case study.

In the remainder of the paper, we first define the notion of code examples and differentiate related concepts. Afterward, we describe the research approach followed by presenting identified API documentation challenges. The next section entails an overview of all best practices candidates for examples in public Web API documentation. We evaluate a subset of eight of these best practices candidates in a case study described in the subsequent section. Finally, we conclude the research paper with a description of the limitations and a conclusion.

2 CODE EXAMPLES

We adopt (Robillard and DeLine, 2011) definition of code examples in API documentation as *"[...] listings, of various length, that show an API being used"*. Thus, an example helps developers to understand how to use API elements in a programming context (Watson, 2012; Jiang et al., 2007). There are four different types of code examples, which are code snippets, tutorials, samples, and production code (Robillard and DeLine, 2011).

- The smallest type of examples are *code snippets*, which show just one aspect of an APIs basic functionality (Robillard and DeLine, 2011; Watson, 2012). Similarly, (Watson, 2012) describes code snippets as short samples of code that show how to use one API element (Watson, 2012).
- A *tutorial* is a sequence of snippets that implement a specific, non-trivial functionality in a step-by-step manner (Robillard and DeLine, 2011; Watson, 2012). A getting started guide is a type of tutorial that enables developers to implement a basic usage of the API (Inzunza et al., 2018). Also, a concept which we equate with tutorials are *recipes*. (Meng et al., 2019) describes recipes as code examples in a cookbook-like fashion. Each recipe describes how to reach a specific solution given a specific problem (Meng et al., 2018).
- Next, a *sample* is a small and self-contained application (Robillard and DeLine, 2011). In comparison to tutorials that show just one functionality, sample apps demonstrate complete and usable programs comprising several features and potentially additional functionality, e.g., a user interface or error handling (Watson, 2012; Nykaza et al., 2002).
- Finally, API consumers can inspect *production code* for code examples (Robillard and DeLine, 2011). For instance, consumers can extract such code examples from open source systems.

An example does not only consist of code but also entails accompanying explanations (Robillard and DeLine, 2011). These explanations provide information that allows consumers to understand the example code and thus modify it (Nasehi and Maurer, 2010; Nasehi et al., 2012; Ko and Riche, 2011;

Meng et al., 2018; Uddin and Robillard, 2015; Thayer et al., 2021; Glassman et al., 2018). Such an explanations can, e.g., comprise a rationale that explains how the code relates to concepts and execution facts (Thayer et al., 2021). The explanations can be incorporated into code in the form of comments (Nasehi et al., 2012) or accompany the example code as textual descriptions (Meng et al., 2018).

Moreover, we distinguish the concept of examples from similar concepts in literature. First, we want to distinguish examples from API *usages*. An API usage captures what an API can be used for (Jiang et al., 2007; Stylos et al., 2009). Hence, examples can demonstrate how to realize usages, i.e., recommended or required sequences of API calls that implement a specific functionality (Jiang et al., 2007). However, an API provider can also use UML2 sequence diagrams to convey usage scenarios as part of API documentation (Jiang et al., 2007).

Next, (Thayer et al., 2021) introduces *usage patterns* as a type of knowledge that consumers require to be able to use an API successfully. A usage pattern is a code pattern describing how calls to several APIs should be coordinated and are accompanied by a rationale that explains how the code relates to concepts and execution facts (Thayer et al., 2021). A usage pattern can be a code snippet. Thus, usage patterns convey what a developer can do with an API (Thayer et al., 2021).

Finally, (Hoffman and Strooper, 2000) and (Hoffman and Strooper, 2003) introduce the idea of using *test cases* as examples in documentation. A test case comprises preconditions, inputs, and expected results to determine if a system meets specific test objectives (ISO/IEC/IEEE 29119-1, 2013). Test cases are executable examples with expected outputs that guarantee precision, completeness, and machine processability.

As a result, we understand examples as code comprising API requests and expected responses with accompanying explanations that demonstrate how to use an API to realize one or more functionalities. Furthermore, we differentiate between code snippets that show one call or functionality and tutorials that present sequences of calls that build on each other. Moreover, samples are self-contained, exemplary applications, while production code showcases API usage in productive applications. Also, an API usage captures what an API can be used for (Jiang et al., 2007; Stylos et al., 2009), and examples can demonstrate how to realize usages.

3 RESEARCH APPROACH

This section describes the research approach for identifying challenges of creating and maintaining API documentation from a provider perspective, identifying best practices candidates for API documentation, and evaluating a subset of eight of these best practices candidates in the context of public Web API documentation. The research approach comprises a literature review, expert interviews, and a case study.

3.1 Literature Review

We reviewed existing literature on API documentation following an approach inspired by (Webster and Watson, 2002) to extract best practices candidates for the design and integration of code examples into Web API documentation that improve consumers' API learning. As a starting point, we conducted an extensive search of the databases ScienceDirect, ACM, IEEE Xplore, and Scopus. Afterward, we uncovered additional relevant publications using forward and backward search. As a result, we collected 17 research papers that present implications, principles, or observations for code examples in the documentation of local APIs, non-public Web APIs, and public Web APIs. We include best practices candidates derived from literature concerned with non-public Web APIs and local APIs since we assume they could also apply to public Web APIs. Still, their applicability first needs to be validated. From the 17 research papers, we derive 32 best practices candidates for designing code examples in public Web API documentation.

3.2 Expert Interviews

Next, we aim to identify challenges related to creating and maintaining API documentation from an API provider perspective and requirements for designing valuable code examples in public Web API documentation using semi-structured expert interviews. Overall, we interviewed 13 professionals, including product owners, architects, and software developers employed by a large multinational software vendor. The prerequisites for participation in the interviews were that each interviewee has professional experience with API provision or consumption and was actively working on a project concerned with API design and documentation at the time of the research endeavor. The interviews took place between April and July 2019 and lasted 39 minutes on average. We audio-recorded 13 interviews and transcribed the relevant parts[1]. We provide an overview of the interviewees, including an ID, a role description, their years of professional experience in software design, and the duration of the interviews, in Tab. 1.

We analyzed the data using open coding, selective coding, and constant comparison as described by (Wiesche et al., 2017). The analysis of the interviewees yields 34 best practices candidates, of which 20 support best practices candidates previously identified in the literature. Therefore, we identify 46 unique best practices candidates from literature and expert interviewees.

3.3 Case Study

Finally, we aim to evaluate the effect of eight best practice candidates for code examples in public Web API documentation on consumer perception and productivity. To reach this goal, we conduct an embedded single case study (Yin, 2013) with a large software vendor.

As a basis we create two different versions of documentation for the existing open-source system Compass[2]. The system enables the integration and monitoring of application landscapes consisting of internal applications running on specific computing clusters and external applications. A user interacts with the system through a GraphQL[3] API.

Overall, the documentation consists of three components, which are a textual description of the API including examples, a GraphQL playground[4], and an interactive specification generated with graphdoc[5]. The textual description references the GraphQL playground and the specification.

We focus on the textual description of the API since it entails the examples, and design two significantly different versions of it; a basic version and an advanced version. However, there is no consensus on the best structure of textual API documentation in literature or practice. Therefore, we derive the structure of the basic textual documentation from leading API management tool providers, e.g., MuleSoft and Apigee (Pillai et al., 2021), and practice-driven guides for technical writers (Johnson, 2021). As a result, we structure the textual documentation into an overview, getting started, tutorial, samples [6], and glossary part.

Since this research aims to analyze best practices candidates for code examples, the tutorial and sample section of the textual descriptions differ between the two versions. More specifically, we realize a set of eight best practices candidates in the tutorial and sample sections of the advanced version of the documentation, which we do not implement in the basic version. We chose the specific set of eight best practices candidates for one of three reasons. Either, previous literature does not mention or only weakly support a best practice candidate described by the interviewees. In addition, we chose best practices candidates for which we identified contradicting statements about their impact on the API consumers' productivity and perception of APIs. Finally, we selected best practices candidates, which we derived from literature that is not specific to public Web APIs.

The basic documentation enables API consumers to use the API by providing only necessary and common documentation sections. The tutorial of the basic documentation describes an exemplary usage scenario consisting of three code snippets of low complexity within a specific context. The samples are four independent code snippets covering essential ways to query Compass elements without context.

The advanced textual documentation builds on the basic documentation by implementing the best practices candidates described in the following.

First of all, the tutorial in the advanced documentation presents a "main scenario" that covers the typical use of the system (BP27). Moreover, the tutorial integrates a button that imports the complete tutorial into the Postman[7] application, thus enabling to easily try-out the examples (BP46). The tutorial description is very concise and problem-oriented (BP30). Furthermore, the tutorial entails seven code snippets with increasing complexity (BP03). The tutorial also follows a storyline with a clear outcome (BP28). Lastly, the tutorial presents alternatives to solve a problem, i.e., includes junctions in the solution path (BP13).

Focusing on the samples, the code snippets in the advanced textual documentation frequently reference the specification (BP38). Again, the samples provide integrated Postman tool support (BP46). Another significant difference is that the samples cover a higher amount of usages by providing eight code snippets, including snippets with higher complexity (BP01, BP03).

In addition, the advanced textual description en-

[1]Due to a malfunction of the recording device, we did not record the interview with I3. Instead, we analyze the field notes we made during the interview.

[2]https://github.com/kyma-incubator/compass

[3]https://graphql.org/

[4]https://github.com/graphql/graphql-playground

[5]https://github.com/2fd/graphdoc

[6]We named the section "samples" since it is common to name a section containing code snippets "samples" or "examples" and we did not want to confuse the case study participants. However, following the definition of (Robillard and DeLine, 2011), the section contains code snippets.

[7]https://www.postman.com/

Table 1: Overview of interview experts, including a reference ID, interviewees' roles, years of experience, and the interview durations.

ID	Role	Experience [years]	Duration [hh:mm]
I1	Software Architect	12	01:15
I2	Product Owner	15	00:32
I3	Enterprise Architect	12	-
I4	Enterprise Architect	10	00:30
I5	Software Architect	29	00:36
I6	Enterprise Architect	7	00:44
I7	Product Owner	12	00:45
I8	Lead Developer	20	01:05
I9	Senior Developer	15	01:31
I10	Senior Developer	6	00:40
I11	Developer	2	00:36
I12	Senior Developer	8	00:41
I13	Senior Developer	20	00:38

tails a "best practices" section. These best practices make it easier for users to use the playground, present beneficial approaches to navigate the documentation components, and provide hints on how to interact with GraphQL APIs.

Overall, 12 professional software developers from the industry partners organization participated in the evaluation. We admitted only participants with at least four years of professional experience into the study. We split the participants into two groups with the aim to balance the mean regarding years of experience across the groups and assigned each group one version of the documentation. An overview of the case study participants is provided in Tab. 2.

Finally, we used multiple approaches to collect case study data, comprising observations, a SUS questionnaire, and open questions to collect quantitative and qualitative data.

After starting the documentation application, setting up the GraphQL playground, and answering organizational questions, the case study setting allowed participants to first read and get acquainted with the assigned version of the API documentation. Next, we confronted the participant with three tasks. The tasks resemble real API usages, have increasing difficulty levels, and are solvable in a limited time. We encouraged the participants to express think-aloud comments. One researcher observed each participant and used a field protocol to capture the time, the number of issued API requests, and the usage of documentation sections during the task solution.

After solving or trying to solve the tasks, we asked the participant to fill in a SUS questionnaire according to (Brooke et al., 1996). Finally, we used open questions to inquire about the perceived usefulness of the documentation, including the most useful, least useful, and missing parts of the documentation. The whole evaluation process was audio recorded.

4 CHALLENGES

The goal of documentation is to enable API consumers to understand and use an API. Therefore, we first present four significant challenges related to documentation from an API provider perspective as perceived by the interviewees.

Unknown Consumer Needs. The first major challenge related to API documentation is that API providers lack information about the API consumers' needs (I2, I3, I4, I5, I8). The API provider does not know who will use the API and for what purpose (I4, I7, I8). Moreover, API consumers regularly want to use APIs for purposes not intended or expected by the API provider (I5, I7), as noted by I7: *"In addition, we often have consumers using the API for something it was not intended for. This is also due to the fact that the provider knows too little about the various use cases of its consumers [...]."*

Coverage. The second challenge concerns the coverage of documentation. API consumers expect the documentation to cover all information necessary to solve their respective problems. As (Meng et al., 2018) highlights, completeness of documenta-

Table 2: Overview of case study participants, including a reference ID, the participants' API-specific experience, and their general professional development experience.

Group A			Group B		
ID	**API Experience**	**Total Experience**	**ID**	**API Experience**	**Total Experience**
A1	10	14	B1	7	9
A2	3	4	B2	7	7
A3	10	15	B3	10	15
A4	4	4	B4	2	4
A5	3	9	B5	6	10
A6	9	10	B6	3	4
Mean	6.50	9.33	**Mean**	5.83	8.17

tion is one of the most important criteria for developers. However, documentation cannot cover all information (I5, I8, I9, I10) due to API consumers having different goals, learning styles, and tool preferences (I8, I9, I10). Also, the provider might not know all possible states of an API, especially for complex APIs or flows across several APIs (I5, I8). Hence, the interviewees report frequently running into issues not covered in the documentation (I2, I4). In addition, according to (Maalej and Robillard, 2013), documentation length increases with the number of included knowledge types, potentially leading to huge documentation. However, API consumers perceive extensive documentation negatively, as I7 highlights *"[...] it can lead to an information overload. There may be everything in it but if it is hard to understand and too long, then it is [...] even counterproductive."* Hence, API providers face the challenge of balancing the coverage of varying information needs and keeping the documentation as concise as possible.

Effortful Creation and Maintenance. Next, the creation and maintenance of documentation is difficult and creates effort for the API provider (I8, I9, I12) (Robillard and DeLine, 2011; Hoffman and Strooper, 2000; Ko et al., 2007; Uddin and Robillard, 2015; Sohan et al., 2017; Mar et al., 2011) as I8 describes: *"And in any case, [creating] these scenarios also involve a certain amount of effort. The question is always whether the teams or the company are willing to make this effort. You can already see that it often fails for the simplest type of documentation, i.e., the specification."* In addition, the API provider needs to update the documentation in case of changes of the API or underlying systems (I1, I2, I8, I9, I11, I13). Up-to-dateness is a major requirement for documentation according to developers (Meng et al., 2018; Robillard and DeLine, 2011), since erroneous documentation leads to frustrated API consumers (Hosono et al., 2019), and potentially even to illegal and unsafe API usages (Zhong and Su, 2013). Hence, API provider organizations should invest in overcoming the perception of documentation as a side product (I1, I3, I5, I8) and motivate internal developers who prefer to write code to create and maintain up-to-date documentation (I1, I8).

Missing Governance. Finally, the interviewees mentioned that many different documentation styles exist for API documentation, making it difficult to know how to design the documentation (I4, I9, I13). Usually, internal guidelines and governance for creating Web API documentation are missing (I6).

5 BEST PRACTICES CANDIDATES

In this section, we present best practices candidates for code examples in API documentation derived from the literature review and the expert interviews. Overall, we identified 46 best practices candidates, as presented in Tab. 3.

The identified best practice candidates span different topics and levels of abstraction. Therefore, some best practice candidates relate to each other or are overlapping. In addition, some best practice candidates are contradictory. Furthermore, most of the existing research on code examples as part of API documentation were conducted in the context of local APIs and not publicly accessible Web APIs. Therefore, we aim to evaluate which best practice candidates are actual best practices applicable for public Web APIs in the next section.

6 VALIDATION OF BEST PRACTICES

We analyze the collected data to evaluate the effect of eight best practices candidates applied to the code snippets and tutorials on the productivity and perception of API consumers using a public Web API. First, we describe the findings of the quantitative analysis

Table 3: Best practices candidates derived from literature and expert interviewees.

ID	Best Practices Candidates	Literature Sources	Inverviewees
BP01	Examples should cover all common usage scenarios of an API.	(Nasehi and Maurer, 2010; Ko et al., 2004)	I5, I7, I8, I9
BP02	Examples should comprise a concise unit of functionality that can be combined into more complex functionality.	(Meng et al., 2018; Robillard and DeLine, 2011; Nasehi et al., 2012)	I8
BP03	Documentation should present examples with different, e.g., increasing, levels of complexity to meet the needs of consumers with different expertise.	(Nasehi et al., 2012; Nykaza et al., 2002)	I9, I11, I12
BP04	Examples should demonstrate how to coordinate sequential requests to a single API to implement (more complex) functionality.	(Nasehi and Maurer, 2010; Robillard, 2009; Sohan et al., 2017)	I1, I2, I4, I8, I12
BP05	Examples should demonstrate how to coordinate requests to multiple APIs to implement (more complex) functionality.	(Robillard and DeLine, 2011; Thayer et al., 2021)	I2, I13
BP06	Examples need to be correct and up-to-date to not frustrate consumers.	(Meng et al., 2018; Robillard and DeLine, 2011)	I1, I2, I8, I9, I11, I12, I13
BP07	Examples should be copyable and thus complete.	(Meng et al., 2018; Meng et al., 2019; Hoffman and Strooper, 2000; Hoffman and Strooper, 2003)	I1, I8, I9, I11
BP08	Examples should be freely accessible.		I8
BP09	Documentation should use compact and readable test cases as code examples.	(Hoffman and Strooper, 2000; Hoffman and Strooper, 2003)	I3, I4, I8, I9
BP10	Examples should adhere to general and community-specific programming conventions.	(Meng et al., 2018; McLellan et al., 1998)	
BP11	Examples should implement "best practices" specific to an API.	(Robillard and DeLine, 2011; Robillard, 2009)	I8, I9
BP12	Examples need to present correct request syntax and semantics, including all necessary HTTP headers, valid parameter, valid data types, and data formats.	(Sohan et al., 2017)	I4, I7, I8, I9, I11, I13
BP13	Examples should demonstrate alternative solution approaches, including corner cases.		I2, I8, I11, I12
BP14	The API provider should describe APIs capabilities usages at the beginning of the example.	(McLellan et al., 1998)	I12
BP15	Examples need to relate to the overall domain of the API.		I4
BP16	Examples should evolve according to the consumers' needs.	(Nasehi et al., 2012)	
BP17	Documentation should not indicate that an example is old if it still works.	(Robillard, 2009; Robillard and DeLine, 2011)	
BP18	Examples should use familiar domains.	(Nasehi et al., 2012)	
BP19	Examples should also demonstrate "unhappy paths".		I3, I8, I11
BP20	Examples should entail or be accompanied by all information necessary for successful authentication.		I2, I7, I11, I13
BP21	Examples should provide client code in different programming languages.		I4
BP22	An example should describe valid test data.		I13
BP23	Tutorials should have a consistent level of detail across all steps.		I13
BP24	Tutorials should be accompanied by suitable visualizations.		I8

Table 3: Best practices candidates derived from literature and expert interviewees (continued).

ID	Best Practices Candidates	Literature Sources	Inverviewees
BP25	Documentation should include advanced tutorials and applications that transfer information on complex API interactions.	(Robillard, 2009)	
BP26	Tutorials should be structured to show some intermediate result after each step.	(Inzunza et al., 2018)	I13
BP27	Documentation should use tutorials to demonstrate the APIs basic functionality.	(Meng et al., 2018; Nykaza et al., 2002)	I1, I2, I6
BP28	Tutorials should present end-to-end usages following a storyline.		I2, I9, I11
BP29	Documentation should entail a sample app that demonstrates the primary usage(s) of the API.		I1, I6
BP30	Code snippets, tutorials, and their explanations should be as concise and problem-oriented as possible.	(Nasehi et al., 2012)	I1, I7, I10, I12, I13
BP31	Explanations of relevant conceptual knowledge should accompany examples.	(Meng et al., 2018; Meng et al., 2019; Thayer et al., 2021)	I1, I9, I12, I13
BP32	An example should describe the intended usages of the API.		I9
BP33	Example explanations should provide information on pre- and postconditions of API interactions.	(Hoffman and Strooper, 2000; Hoffman and Strooper, 2003)	I12, I13
BP34	Example explanations should provide information on non-deterministic API behavior.	(Hoffman and Strooper, 2000; Hoffman and Strooper, 2003)	
BP35	The explanation should describe shortcomings of the API itself and potential workarounds.	(Nasehi et al., 2012)	
BP36	Explanations accompanying examples should describe limitations of the solution that the example presents.	(Nasehi et al., 2012)	
BP37	Explanations of code examples should reference related or alternative solutions.	(Nasehi et al., 2012)	
BP38	Example explanations need to explain how the code relates to low-level API elements.	(Thayer et al., 2021)	I9
BP39	Examples should transfer information on the API design rationale.	(Nykaza et al., 2002; Robillard, 2009)	
BP40	Explanations should describe the reasons for error messages and measures to solve these errors.		I1, I6, I7, I8, I11
BP41	Explanations should describe the sense of each part of an example request.		I11, I12, I13
BP42	The documentation should make it easy to separate example code from textual descriptions.	(Meng et al., 2019)	
BP43	The documentation should highlight how explanations relate to the code.	(Meng et al., 2019; Meng et al., 2020)	
BP44	The documentation should highlight crucial elements, e.g., the names of relevant API elements or design patterns, in the code examples and explanations.	(Nasehi et al., 2012)	I13
BP45	Examples should present indicators that provide information on code example quality.	(Glassman et al., 2018)	
BP46	The documentation should enable the execution of examples within the documentation or common tools.	(Meng et al., 2019; Jeong et al., 2009)	I2, I3, I6, I8, I9, I10, I11, I12

followed by the results of the qualitative analysis.

6.1 Quantitative Analysis Results

The quantitative analysis comprises metrics on the needed time, success, and documentation feature usage observed during the evaluation. In addition, we present the results of the SUS survey. The sample size for all presented statistics is n = 12.

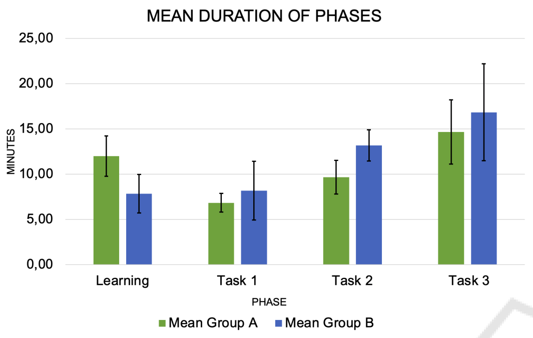

Figure 1: Overview of the time that participants took to initially read the API documentation (learning) and solve the three tasks.

First, we analyze the time that participants took to initially read the documentation and to solve all tasks as presented in Tab. 1. The analysis shows that, overall, participants of group A needed on average 43:10 minutes whereas group B requires 46:00 minutes. Moreover, we observe that while group A spend more time learning about the API, they solved all tasks faster than group B. However, the improvement in time is only statistically significant for the more complex tasks 2 (t=2.29, df=10, p=0.05, one-tailed) and 3 (t=1.91, df=10, p=0.05, one-tailed).

Next, we analyzed the success of the participants solving the tasks. We assigned points to the level of achievement of solutions to do so. A participant receives one point for a solved task, a half-point for an unfinished task with the right approach, and no points for wrong solutions. Group A reached on average 2.92 points and group B reaches 2.83 points. Hence, the assessment yields, that both groups are very successful in solving all three tasks. As a result, we cannot derive any significant differences with regards to the success of solving the tasks between the groups.

We also observed the number of API requests that the participants issue to Compass and their success. Overall, the participants made between 12 and 16 requests to the application during the study. On average, the rate of successful requests is 61% for group A and 39% for group B. Hence, the request success rate for members of group A is significantly higher compared to group B (t=5.66, df=10, p=0.05, one-tailed). Also, the standard deviation differs between the groups with group A yielding a standard deviation of 0.019 and group B with 0.084. Hence, the success rate of members of the group B varied more.

Furthermore, we noted down the approximate fractions of time that each participant spent on the documentation's different sections. In Fig. 2, we visualize the relative usage of each documentation section for each group. We observe that group B used the getting started guide more often than group A. However, the members of group A spend more time in the tutorial and best practices sections. Group A also shows higher variations across most sections, indicating that participants value the documentation sections differently.

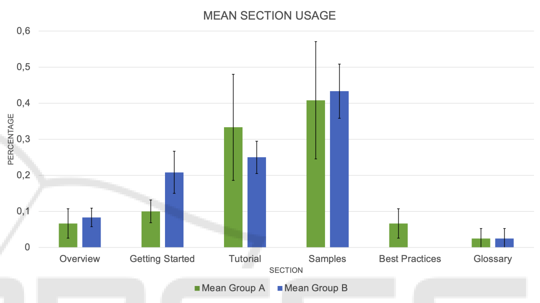

Figure 2: Overview of the relative amount of times the participants consulted specific documentation sections.

Moreover, the participants rated the advanced documentation with an average SUS score of 85.8 points compared to 75 points for the basic documentation. Hence, the software developers perceive the usability of the advanced documentation as better.

Finally, we analyzed the data for correlation between variables. First, we identified a moderately strong correlation (r = 0.79) between the request success rates and the duration of the learning phase, indicating that investment into learning the API results in a higher probability of making successful API requests. In addition, we see a moderate correlation (r = 0.66) between the success rate and the perceived usability measured with the SUS questionnaire. Interestingly, the participants' API and software engineering experience does not seem to correlate with the API request success rate (r = 0.25, r = 0.32).

6.2 Qualitative Results

We derive the qualitative results from the think-aloud protocol and semi-structured interviews with the case study participants. We describe our observations along the questions we asked the participants. Since the participants discussed the documentation in general, we also report observations not related exclusively to examples.

First, we asked the participants if they perceived the API documentation as helpful in solving the tasks. Five participants answered that they do not think of the overview section as beneficial for the case study tasks but it might be helpful for very complex cases or tasks aiming at integration with 3rd applications (A2, A4; B3, B4, B5). In addition, the API specification is useful for looking up more detailed information and targeted problem solving (A2; B5, B6). Also, the execution environment was considered "vital" (B5) and helped participants gain confidence and experience using the API (A2, A3). Both groups used the samples as a lookup point, but group B saw the samples as the main entry point (B1, B2, B5, B6). The tutorial section was perceived as helpful by participants of group A because they could solve tasks by merely adjusting the code examples (A1, A2). Moreover, the advanced documentation puts the code snippets in the tutorial into a context, and therefore the participants expect it to help with real implementation tasks (A4, A5). However, group B did rarely comment on the tutorials. Finally, participants of group A commented that the tool support for Postman might only be helpful if the API consumers are Postman "power users" (A3, A5).

Next, we asked what parts of the documentation the participants liked the most and why. All respondents stated that the specification and playground were among their favorite parts of the documentation. The specification supported the participants with clickable elements (A1, A2), a search function (B3, B4), and a structured presentation (B5). The reason for the positive perception of the playground was that it allowed consumers to try out code snippets and tutorials (A1, A2, A4; B4, B6). Group A also liked the tutorials, primarily because of the many references to other resources included in the advanced version (A1, A2, A5). Moreover, participants of group A stated that they liked the coverage of usages with examples (A3, A4). In comparison, participants of group B appreciated the samples the most (B1, B2, B4) since they provided an entry point into solving the tasks (B1, B2). At last, two participants of group A positively mentioned the best practices section, explaining that such information is often missing in documentation (A4, A5). The best practices were only available to group A.

The third question we asked the case study participants was which parts of the documentation they did not like at all and why. All participants agreed that the overview and glossary were the most useless sections because they presented too much information that was unnecessary for solving the tasks (A2, A3, B2). However, the provided conceptual information might be necessary for more complex scenarios or non-technical stakeholders (A4; B4). Also, the participants criticized that we did not highlight important aspects of the text in the documentation enough (A5; B2, B5). A complaint specific to group B was that the descriptions of the samples and the tutorial were too long (B1, B6). In general, the participants of group B perceived the documentation as not "developer-centric" enough (B3, B4).

Lastly, we asked the participants which features they missed the most in the documentation. Several participants mentioned the lack of "helper buttons", e.g., buttons that automatically copy a code snippet or directly transfer snippets into the playground for execution (A5; B2, B5). Also, one participant complained about missing information on prerequisites (A4). Most statements about missing features came from group B. Most importantly, they requested more samples with higher complexity (B1, B2, B6). Also, group B participants criticized that they lacked links between resources, which required them to conduct more searches (B2, B6). Finally, one participant requested more concise and practical descriptions of the tutorial and the samples (B4).

As a result, we verify the best practice candidates BP01, BP03, BP13, BP27, BP28, BP30, BP38, BP46 to positively influence the consumers perception of an API, thus validating them as best practices for public Web API documentation.

7 LIMITATIONS

Overall, we interviewed 13 experts and recruited 12 participants for the case study. Due to the low number of participants, the statistical validity of the findings is limited. Also, a larger sample of case study participants might have led to additional findings. However, all interview experts and case study participants are professional developers with several years of experience. Therefore we believe the results of this study to present valid insights. Nevertheless, future studies should also investigate the validity of the best practices for API consumers with less experience to take into account the varying levels of experience that API consumers might have.

Moreover, all participants are employees of a single industry partner organization that provides several publicly accessible Web APIs. Hence, these developers might be more interested in API documentation than the average professional developer.

Another limitation arises from the Web API we chose for the case study. The Compass API is embedded in a complex domain and is a GraphQL API. APIs of different complexity and using other technology might have different documentation requirements.

Finally, we also realized several best practices at once as part of the advanced documentation and added a best practices section. Therefore, it is not possible to isolate the effect of a single best practice candidate. Hence, future research could investigate the effect of the realization of each single best practices.

8 CONCLUSION

Learning new APIs is an every-day task for developers (Meng et al., 2018; Meng et al., 2019; Glassman et al., 2018) and it is well known, that documentation is a crucial learning resource for API consumers (Robillard, 2009; Lethbridge et al., 2003; McLellan et al., 1998). Nevertheless, consumers report that API documentation only rarely meets their needs (Meng et al., 2018).

Therefore, we first derive four significant challenges API providers face when creating public Web API documentation from 13 expert interviews. These challenges are unknown customer needs, the difficulty of balancing the coverage of varying information needs and keeping documentation concise, the high effort of creating and maintaining documentation, and missing guidance and governance. Each of these challenges provides a starting point for future research.

Also, code examples comprising code snippets, tutorials, sample apps, and productive code (Robillard and DeLine, 2011) play a crucial role in API documentation (Nykaza et al., 2002; Meng et al., 2018; Meng et al., 2019; Robillard, 2009; Ko et al., 2007; Nasehi and Maurer, 2010; McLellan et al., 1998; Meng et al., 2020; Jeong et al., 2009; McLellan et al., 1998). Nevertheless, examples have to meet specific quality criteria to unlock their potential and support the learnability of APIs (Meng et al., 2018; Robillard and DeLine, 2011; Robillard, 2009; Nykaza et al., 2002). Hence, we identify 46 best practices candidates for examples in public Web API documentation in literature and expert interviews. In addition, we choose a subset of eight best practices (BP01, BP03, BP13, BP27, BP28, BP30, BP38, BP46) and analyze their impact on API consumers' productivity and perception when learning a new API. As a result, we can verify that the chosen best practices candidates are actual best practices for examples in public Web API documentation.

Future work should analyze the impact of other best practices candidates for examples in public Web API documentation on consumer perception and productivity.

ACKNOWLEDGMENTS

We want to thank the participants of the interviews and the case study for their collaboration and the interesting insights.

REFERENCES

Brooke, J. et al. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7.

Glassman, E. L., Zhang, T., Hartmann, B., and Kim, M. (2018). Visualizing api usage examples at scale. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12.

Hoffman, D. and Strooper, P. (2000). Prose+ test cases= specifications. In *Proceedings. 34th International Conference on Technology of Object-Oriented Languages and Systems-TOOLS 34*, pages 239–250. IEEE.

Hoffman, D. and Strooper, P. (2003). Api documentation with executable examples. *Journal of Systems and Software*, 66(2):143–156.

Hosono, M., Washizaki, H., Honda, K., Nagumo, H., Sonoda, H., Fukazawa, Y., Munakata, K., Nakagawa, T., Nemoto, Y., Tokumoto, S., et al. (2019). Inappropriate usage examples in web api documentations. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 343–347. IEEE.

Inzunza, S., Juárez-Ramírez, R., and Jiménez, S. (2018). Api documentation. In Rocha, Á., Adeli, H., Reis, L. P., and Costanzo, S., editors, *Trends and Advances in Information Systems and Technologies*, pages 229–239, Cham. Springer International Publishing.

ISO/IEC/IEEE 29119-1 (2013). Iso/iec/ieee 29119-1: Software and systems engineering — software testing — part 1: Concepts and definitions — first edition 2013-09-01. Technical report, ISO.

Jeong, S. Y., Xie, Y., Beaton, J., Myers, B. A., Stylos, J., Ehret, R., Karstens, J., Efeoglu, A., and Busse, D. K. (2009). Improving documentation for esoa apis through user studies. In Pipek, V., Rosson, M. B., de Ruyter, B., and Wulf, V., editors, *End-User Development*, pages 86–105, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jiang, J., Koskinen, J., Ruokonen, A., and Systa, T. (2007). Constructing usage scenarios for api redocumentation. In *15th IEEE International Conference on Program Comprehension (ICPC'07)*, pages 259–264. IEEE.

Johnson, T. (2021). Documenting apis: A guide for technical writers and engineers. https://idratherbewriting.com/learnapidoc/. (accessed 13.12.2021).

Ko, A. J., DeLine, R., and Venolia, G. (2007). Information needs in collocated software development teams. In *29th International Conference on Software Engineering (ICSE'07)*, pages 344–353. IEEE.

Ko, A. J. and Riche, Y. (2011). The role of conceptual knowledge in api usability. In *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 173–176. IEEE.

Lethbridge, T. C., Singer, J., and Forward, A. (2003). How software engineers use documentation: The state of the practice. *IEEE software*, 20(6):35–39.

Maalej, W. and Robillard, M. P. (2013). Patterns of knowledge in api reference documentation. *IEEE Transactions on Software Engineering*, 39(9):1264–1282.

Mar, L. W., Wu, Y.-C., and Jiau, H. C. (2011). Recommending proper api code examples for documentation purpose. In *2011 18th Asia-Pacific Software Engineering Conference*, pages 331–338. IEEE.

McLellan, S. G., Roesler, A. W., Tempest, J. T., and Spinuzzi, C. I. (1998). Building more usable apis. *IEEE software*, 15(3):78–86.

Meng, M., Steinhardt, S., and Schubert, A. (2018). Application programming interface documentation: what do software developers want? *Journal of Technical Writing and Communication*, 48(3):295–330.

Meng, M., Steinhardt, S., and Schubert, A. (2019). How developers use api documentation: an observation study. *Communication Design Quarterly Review*, 7(2):40–49.

Meng, M., Steinhardt, S. M., and Schubert, A. (2020). Optimizing api documentation: Some guidelines and effects. In *Proceedings of the 38th ACM International Conference on Design of Communication*, SIGDOC '20, New York, NY, USA. Association for Computing Machinery.

Nasehi, S. M. and Maurer, F. (2010). Unit tests as api usage examples. In *2010 IEEE International Conference on Software Maintenance*, pages 1–10. IEEE.

Nasehi, S. M., Sillito, J., Maurer, F., and Burns, C. (2012). What makes a good code example?: A study of programming q&a in stackoverflow. In *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, pages 25–34. IEEE.

Nykaza, J., Messinger, R., Boehme, F., Norman, C. L., Mace, M., and Gordon, M. (2002). What programmers really want: results of a needs assessment for sdk documentation. In *Proceedings of the 20th annual international conference on Computer documentation*, pages 133–141.

Pillai, S., Iijima, K., O'Neill, M., Santoro, J., Jain, A., and Ryan, F. (2021). Magic quadrant for full life cycle api management. Technical report, Gartner, CT, USA.

Robillard, M. P. (2009). What makes apis hard to learn? answers from developers. *IEEE software*, 26(6):27–34.

Robillard, M. P. and DeLine, R. (2011). A field study of api learning obstacles. *Empirical Software Engineering*, 16(6):703–732.

Sohan, S., Maurer, F., Anslow, C., and Robillard, M. P. (2017). A study of the effectiveness of usage examples in rest api documentation. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 53–61. IEEE.

Stylos, J., Faulring, A., Yang, Z., and Myers, B. A. (2009). Improving api documentation using api usage information. In *2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 119–126. IEEE.

Thayer, K., Chasins, S. E., and Ko, A. J. (2021). A theory of robust api knowledge. *ACM Transactions on Computing Education (TOCE)*, 21(1):1–32.

Uddin, G. and Robillard, M. P. (2015). How api documentation fails. *Ieee software*, 32(4):68–75.

Watson, R. B. (2012). Development and application of a heuristic to assess trends in api documentation. In *Proceedings of the 30th ACM international conference on design of communication*, pages 295–302.

Webster, J. and Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS quarterly*, pages xiii–xxiii.

Wiesche, M., Jurisch, M. C., Yetton, P. W., and Krcmar, H. (2017). Grounded theory methodology in information systems research. *MIS Q.*, 41(3):685–701.

Yin, R. K. (2013). *Case study research: Design and methods*, volume 5. Sage Publications, Los Angeles, CA.

Zhong, H. and Su, Z. (2013). Detecting api documentation errors. In *Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications*, pages 803–816.